1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт компьютерных наук и кибербезопасности

**Высшая школа кибербезопасности**

**ЛАБОРАТОРНАЯ РАБОТА**

1. «Создание базы данных на языке Си»
2. по дисциплине «Структуры данных»
3. Выполнил
4. Кондратьев Д. О., студент гр. 5131001/30003 \_\_\_\_\_\_\_\_\_\_\_\_

<*подпись*>

1. Преподаватель
2. Семьянов П. В., старший преподаватель \_\_\_\_\_\_\_\_\_\_\_\_

<*подпись*>

1. Санкт-Петербург
2. 2024

СОДЕРЖАНИЕ

[Цели и задачи 2](#_Toc162825164)

[Введение 2](#_Toc162825165)

[Ход работы 3](#_Toc162825166)

[Результаты 5](#_Toc162825167)

[Вывод 5](#_Toc162825168)

[Приложение 6](#_Toc162825169)

# Цели и задачи

Цель работы: научиться работать с динамической памятью.

Задача: сделать простейшую безразмерную базу данных.

# Введение

Так как база данных должна быть безразмерной, то необходимо использовать динамическую структуру данных. При выборе между динамическим массивом и линейным списком, разумнее будет выбрать линейный список, так как в случае использования динамического массива придётся выделять память при расширении базы данных при помощи функции realloc(). Минусами такого решения являются медленная работа функции realloc(), а также отсутствие аналогичной функции во многих языках программирования, что делает наше решение не универсальным.

Таким образом выбор был сделан в пользу линейного списка.

Данный отчет посвящен рассмотрению принципов построения базы данных на основе линейных списков в языке программирования C. Мы рассмотрим основные операции добавления, удаления и поиска в базе данных на основе линейного списка, а также приведем примеры кода для их реализации.

# Ход работы

Для начала создадим структуру car, которая будет содержать информацию об автомобилях. В ней будут определены переменные типа char, int, enum, union и float. Для хранения названия автомобиля, количества лошадиных сил, типа двигателя, живучести (для автомобилей с ДВС и гибридов расход топлива, для электромобилей – запас хода) и разгон от 0 до 100 соответственно.

Затем напишем функцию create\_list, которая будет создавать и возвращать ссылку на первый элемент линейного списка, на основе которого затем будет строиться база данных.

Теперь база данных готова принимать автомобили. Напишем функцию add, которая будет принимать все необходимые параметры для сохранения информации об автомобиле. Функция, начиная с начала линейного списка, доходит до конца, создает новую структуру и, выделяя память с помощью функции malloc(sizeof(struct car)), вставляет новый автомобиль с заданными параметрами в конец списка.

Так как требуется реализовать считывания базы данных из файла, напишем функцию input, которая будет построчно читать заданный текстовый файл, считывая из него информацию об автомобилях и затем вызывать функцию add для каждого из них.

A diagram of a car

Description automatically generatedРеализуем функцию delete, удаляющую автомобиль по заданному названию. Функция будет удалять все элементы, в которых переданная строка является подстрокой названия автомобиля, поэтому с этой функцией нужно быть аккуратным. Функция идёт с начала линейного списка и, если встречается элемент, строка из поля model которого содержит в себе запрос, удаляет элемент и очищает память при помощи функции free(cur\_car).

Функция db\_output будет проходиться по линейному списку и вызывать вспомогательную функцию output, которая выводит данные об автомобиле в приятном для человека формате:

A screenshot of a computer

Description automatically generated

Также реализуем функцию поиска search, которая, подобно функции delete, будет выводить все позиции, в названия которых входит передаваемая строка.

Для удобной работы с утилитой реализуем систему команд, с помощью которых можно будет удалять, добавлять, искать и печатать элементы базы данных:

* с помощью команды help можно будет ознакомиться со списком имеющихся команд
* add – добавить автомобиль в базу данных
* input – загрузить в базу данных автомобили из файла
* output – вывод информации обо всех автомобилях из базы данных
* delete – удаление автомобилей
* search – поиск по базе данных
* quit/exit/q - завершить работу приложения

В случае ввода неизвестной команды программа оповещает об этом и предоставляет возможность исправиться:

A white background with black text

Description automatically generated

# Результаты

В результате работы была реализована программа для работы с базой данных на языке Си. Создаваемая база данных является безразмерной и содержит в себя все основные типы языка Си, включая enum и union. Поддерживаются операции считывания из файла, удаления элеманта, вставки, вывода и поиска. Взаимодействие с программой реализовано посредством командной строки.

# Вывод

В ходе работы я научился работать с динамической памятью.

# Приложение

#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
  
  
struct car {  
 char model[50];  
 int power;  
 enum {*ICE*, *ELECTRIC*, *HYBRID*} engine\_type;  
 union {  
 float fuel\_cons;  
 int range;  
 } durability;  
 float acceleration0\_100;  
 struct car \*next;  
};  
  
  
const char\* engine\_type\_to\_str(struct car \*object)  
{  
 switch (object -> engine\_type)  
 {  
 case *ICE*: return "ICE";  
 case *ELECTRIC*: return "Electric";  
 case *HYBRID*: return "Hybrid";  
 }  
}  
  
int str\_to\_engine\_type(const char\* s) {  
 if (strcmp(s, "ICE") == 0) return 0;  
 else if (strcmp(s, "Electric") == 0) return 1;  
 else if (strcmp(s, "Hybrid") == 0) return 2;  
 else return -1;  
}  
  
struct car \*create\_list() {  
 *// Выделение памяти под корень списка* struct car \*tmp = malloc(sizeof(struct car));  
 *// Присваивание значения узлу* for (int i = 0; i < 4; i++) tmp -> model[i] = "head"[i];  
 *// Присваивание указателю на следующий элемент значения NULL* tmp -> next = **NULL**;  
 return(tmp);  
}  
  
void add(struct car \*head, const char \*s, int power, float acceleration, int engine\_type, float durability) {  
 struct car \*tail;  
 tail = head;  
 while (tail -> next != **NULL**) {  
 tail = tail -> next;  
 }  
 struct car \*tmp = malloc(sizeof(struct car));  
 int i = 0;  
 while (s[i] != '\0' && s[i] != '\n') {  
 tmp -> model[i] = s[i];  
 i++;  
 }  
 tmp -> power = power;  
 tmp -> acceleration0\_100 = acceleration;  
 tmp -> engine\_type = engine\_type;  
 if (tmp -> engine\_type == *ICE* || tmp -> engine\_type == *HYBRID*) tmp -> durability.fuel\_cons = durability;  
 else tmp -> durability.range = durability;  
 tmp -> next = **NULL**;  
 tail -> next = tmp;  
}  
  
void delete(struct car \*head, const char \*model) {  
 struct car \* cur\_car;  
 struct car \* prev\_car;  
 cur\_car = head;  
 int count = 0;  
 while (cur\_car -> next != **NULL**) {  
 prev\_car = cur\_car;  
 cur\_car = cur\_car -> next;  
 if (strstr(cur\_car -> model, model)) {  
  
 prev\_car -> next = cur\_car -> next;  
*// cur\_car = (struct car\*) calloc(1, sizeof(struct car));* free(cur\_car);  
 count += 1;  
 cur\_car = prev\_car;  
 }  
 }  
 printf("%d objects was deleted\n", count);  
}  
  
  
void input(const char \*filename, struct car \*head) {  
 FILE \*fin;  
 fin = fopen(filename, "r");  
 if (!fin) {  
 printf("Data base uploading error\n");  
 return;  
 }  
 char input\_string[50];  
 while (fgets(input\_string, 50, fin) != **NULL**) {  
 char model[50];  
 int i = 0;  
  
 int im = 0;  
 while (input\_string[i] != ':') {  
 model[im] = input\_string[i];  
 i++;  
 im++;  
 }  
 model[im] = '\0';  
 i++;  
  
 int ip = 0;  
 char power\_str[50];  
 while (input\_string[i] != ':') {  
 power\_str[ip] = input\_string[i];  
 i++;  
 ip++;  
 }  
 power\_str[ip] = '\0';  
 int power = strtol(power\_str, **NULL**, 10);  
 i++;  
  
 int ia = 0;  
 char acc\_str[50];  
 while (input\_string[i] != ':') {  
 acc\_str[ia] = input\_string[i];  
 i++;  
 ia++;  
 }  
 acc\_str[ia] = '\0';  
 i++;  
 float acceleration = atof(acc\_str);  
  
 int ie = 0;  
 char str\_engine\_type[50];  
 while (input\_string[i] != ':') {  
 str\_engine\_type[ie] = input\_string[i];  
 i++;  
 ie++;  
 }  
 power\_str[ip] = '\0';  
 i++;  
 int engine\_type = strtol(str\_engine\_type, **NULL**, 10);  
  
  
 int id = 0;  
 char str\_durability[50];  
 while (input\_string[i] != '\0' && input\_string[i] != '\n') {  
 str\_durability[id] = input\_string[i];  
 i++;  
 id++;  
 }  
 str\_durability[id] = '\0';  
 i++;  
 float durability = atof(str\_durability);  
  
 add(head, model, power, acceleration, engine\_type, durability);  
 }  
 printf("Data base uploaded successfully\n");  
}  
  
void output(struct car \* elem) {  
 printf("Car model: %s\n"  
 "Power: %d hp\n"  
 "0-100 acceleration: %.1f s\n" , elem -> model, elem -> power, elem -> acceleration0\_100);  
 printf("Engine type: %s\n", engine\_type\_to\_str(elem));  
 if (elem -> engine\_type == *ICE* || elem -> engine\_type == *HYBRID*) {  
 printf("Fuel consumption: %.1f L/100km\n", elem -> durability.fuel\_cons);  
 }  
 else printf("Range: %d km\n", elem -> durability.range);  
 printf("------------------\n");  
}  
  
void search(struct car \* head, const char \*request){  
 struct car \*cur\_car;  
 cur\_car = head;  
 int flag = 0;  
 printf("Found objects:\n");  
 printf("------------------\n");  
 while (cur\_car -> next != **NULL**) {  
 cur\_car = cur\_car -> next;  
 if (strstr(cur\_car -> model, request)) {  
 flag = 1;  
 output(cur\_car);  
 }  
 }  
 if (flag == 0) printf("-");  
}  
  
  
void db\_output(struct car \* head) {  
 struct car \*cur\_car = head;  
 if (cur\_car -> next != **NULL**) cur\_car = cur\_car -> next;  
 else {  
 printf("-\n");  
 return;  
 }  
 printf("------------------\n");  
 while (cur\_car != **NULL**) {  
 output(cur\_car);  
 cur\_car = (struct car \*) cur\_car -> next;  
 }  
}  
void print\_command\_list() {  
 printf("q/quit/exit - exit\n");  
 printf("input - upload data base\n");  
 printf("output - print entire data base\n");  
 printf("search - search by model\n");  
 printf("add - add car to database\n");  
 printf("delete - delete car from database\n");  
  
}  
  
int main() {  
 struct car \*head = create\_list();  
 char command[50];  
 char filename[50];  
 char request[50];  
 char model[50];  
 char model\_ending[50];  
 int power;  
 float acceleration;  
 char str\_engine\_type[10];  
 float durability;  
  
 while (1){  
 printf("\nEnter command (help - command list):\n");  
  
 scanf("%s", command);  
 if (strcmp(command,"help") == 0) {  
 print\_command\_list();  
 }  
 else if (strcmp(command,"output") == 0) {  
 printf("\nData base output:\n");  
 db\_output(head);  
 }  
 else if (strcmp(command, "input") == 0) {  
 printf("Enter name of data base to upload: \n");  
 scanf("%s", filename);  
 input(filename, head);  
 }  
 else if (strcmp(command,"search") == 0) {  
 printf("Enter search request: \n");  
 scanf("%s", request);  
 search(head, request);  
 }  
 else if (strcmp(command,"add") == 0) {  
 printf("Enter car model: \n");  
 *//char\* model1;* scanf("%s", model);  
 fgets(model\_ending, 50, **stdin**);  
 *//model1 = strcat(model, " ");* printf("Enter car power (hp): \n");  
 scanf("%d", &power);  
 printf("Enter 0-100 acceleration time (s): \n");  
 scanf("%f", &acceleration);  
 str\_engine\_type[0] = '\0';  
 while (str\_to\_engine\_type(str\_engine\_type) == -1) {  
 printf("Enter car engine type: (ICE/Electric/Hybrid)\n");  
 scanf("%s", str\_engine\_type);  
 }  
 if (str\_to\_engine\_type(str\_engine\_type) == *ICE* || str\_to\_engine\_type(str\_engine\_type) == *HYBRID*) {  
 printf("Enter average fuel consumption value (L/100km): \n");  
 }  
 else printf("Enter estimated range value (km): \n");  
 scanf("%f", &durability);  
 add(head, **strcat**(model, model\_ending), power, acceleration, str\_to\_engine\_type(str\_engine\_type), durability);  
 printf("Car added successfully\n");  
 }  
 else if (strcmp(command,"delete") == 0) {  
 printf("Enter model of car to delete: \n");  
 scanf("%s", model);  
 delete(head, model);  
 }  
 else if (strcmp(command,"q") == 0 || strcmp(command,"exit") == 0 || strcmp(command,"quit") == 0) {  
 return 0;  
 }  
 else if (strcmp(command, "f") == 0) input("cars.txt", head);  
 else printf("Unknown command\n");  
 }  
 }